Nizhny Novgorod State University

Institute of Information Technologies, Mathematics and Mechanics

Department of Computer software and supercomputer technologies

Educational course
«Introduction to deep learning
using the Intel® neon™ Framework»

The Intel® nGraph™ overview

Supported by Intel

Valentina Kustikova,
Phd, lecturer, department of Computer software
and supercomputer technologies

Content

- Overview of the Intel® nGraph™
- Key concepts of the Intel® nGraph™
- Building and installing the Intel® nGraph™
- Building and installing the Intel® neon™ Framework for the Intel® nGraph™
- Example of training and testing the model

OVERVIEW OF THE INTEL® NGRAPH™

Intel® nGraph™ (1)

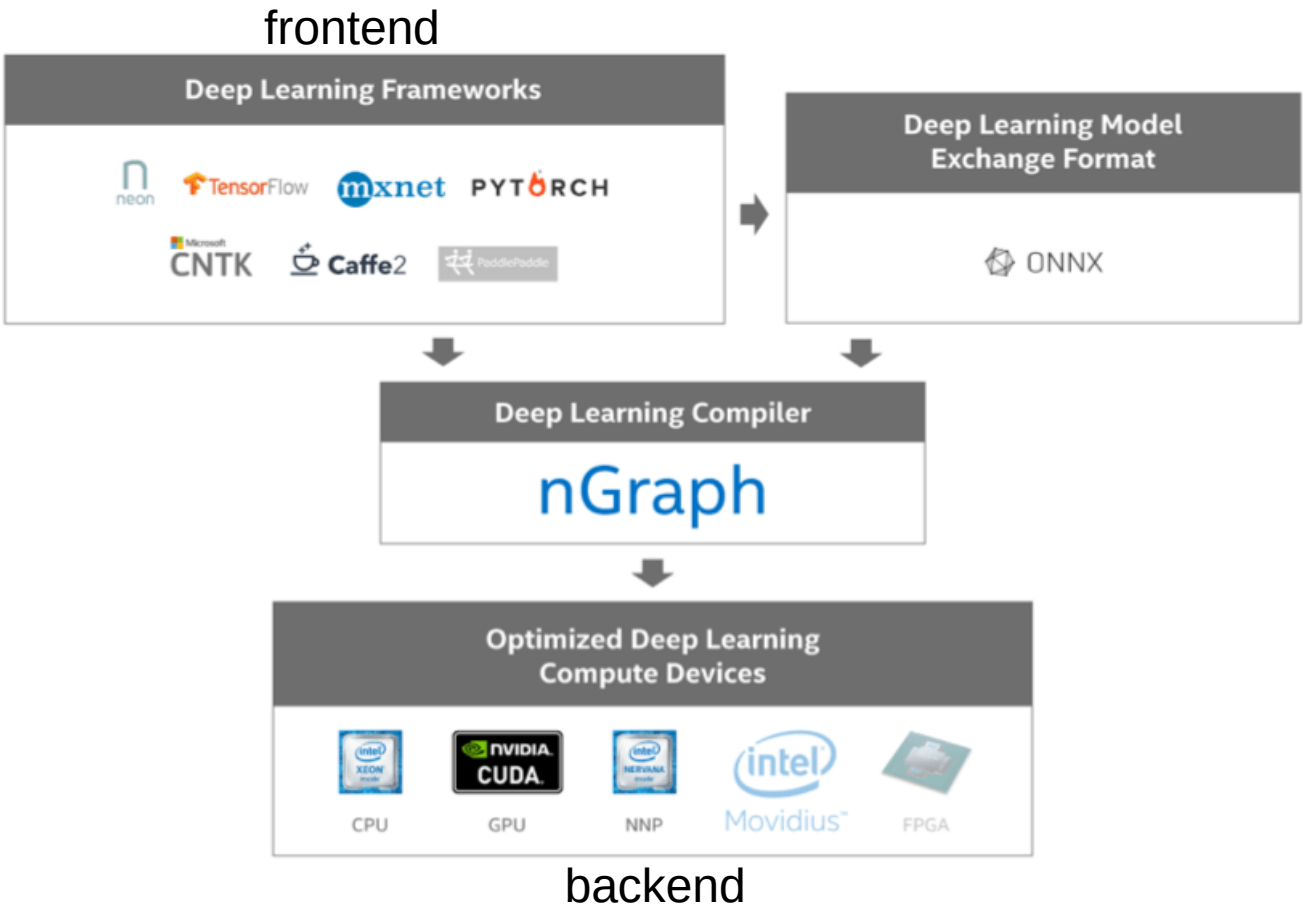
- ***The goal of the nGraph*** is to make the work of data analysts, algorithm developers, software engineers and programmers portable, adaptive and deployable across the most advanced machine learning hardware available today (optimized deep learning devices):
  - Intel® Architecture Processors (CPUs)
  - Intel® Integrated Graphics (GPUs)
  - Intel® Nervana™ Neural Network Processor™ (NNPs)
  - NVIDIA CUDA (GPUs)
  - Field Programmable Gate Arrays (FPGAs)
  - Movidius
- Intel® nGraph™ (nGraph) consists of C ++ library, the runtime and compiler for deep learning systems

Intel® nGraph™ (2)

Models in the format
of different deep learning
libraries

Model compiler
for different hardware

Compute devices



* nGraph [http://ngraph.nervanasys.com/docs/latest].

Schemes of work with the nGraph

- Schemes of work with the nGraph:
  - Building third-party deep learning framework (TensorFlow, MXNet, Intel® neon™ Framework) to work with the nGraph and the subsequent training of deep model
  - Converting the trained model to the format Open Neural Network Exchange Format (ONNX) to execute on the hardware supported by the nGraph

- ***Note:*** in the rest of the lecture a procedure for installing Neon to work with nGraph and an example of training/testing a model are considered

Model portability

- At the moment, three deep learning tools are supported with a pre-optimized environment for learning deep models:
  - TensorFlow
  - MXNet
  - Intel® neon™ Framework

- ***Model portability***
  - The model can be placed near the data, rather than with the core of the deep learning tool
  - There is no need to place data on computing resources used for training

Model adaptability

- **Model adaptability**
  - The trained model can be imported for testing on target servers using nGraph
  - The import of the model is carried out in the open model format ONNX [http://onnx.ai]

- Testing models in ONNX format is provided by an auxiliary tool that has a programming interface in Python [https://github.com/NervanaSystems/ngraph-onnx]
- Frameworks that support the ONNX format:
  - TensorFlow, MXNet, neon, CNTK, PyTorch, Caffe2

Deployment

- ***Deployment***
  - Any modern CPU architecture can be used to train and test deep neural network models
  - The model can be adapted to work on different hardware

- Supported hardware:
  - Intel® Architecture Processors (CPUs)
  - Intel® Nervana™ Neural Network Processor™ (NNPs)
  - NVIDIA GPUs
- Coming soon:
  - Field Programmable Gate Arrays (FPGAs)
  - Movidius [https://www.movidius.com]

How to start?

- Main sources:
  - Intel® nGraph™ documentation [http://ngraph.nervanasys.com/docs/latest]
  - Sources of the Intel® nGraph™ [https://github.com/NervanaSystems/ngraph]
- Additional sources:
  - Intel AI Academy paper [https://ai.intel.com/ngraph-a-new-open-source-compiler-for-deep-learning-systems]
  - Intel® nGraph™. An Intermediate Representation, Compiler, and Executor for Deep Learning [https://arxiv.org/pdf/1801.08058.pdf]
  - Other papers and tutorials

KEY CONCEPTS OF THE INTEL® NGRAPH™

Deep model representation in the nGraph

- The deep model in the nGraph is represented by a ***data-flow graph***
  - The ***nodes*** of the graph correspond to mathematical operations
  - The ***edges*** correspond to multidimensional datasets (tensors)

- ***Note:*** this model representation is supported in TensorFlow, MXNet and Neon, so the nGraph developers provide direct training of the models in the format of the specified frameworks via the Python programming interface

Graph basics (1)

- ***Framework bridges***
  - ***A framework bridge*** is a software layer (typically a plugin for or an extension to a framework) that translates the data science-oriented language into a compute-oriented language called a ***data-flow graph***
  - ***The bridge*** can then present the problem to the nGraph as abstraction layer which is responsible for execution on an optimized backend
  - Optimization of execution is carried out by replacing certain subgraphs of computations with more optimal ones
  - ***The bridge is a plug-in or an extension of a deep learning framework that allows you to convert a model into nGraph format***

Graph basics (2)

- ***Transformer ops***
  - A framework bridge may define its own bridge-specific ops supported by the deep learning framework
  - Bridge-specific ops can be converted to transformer ops

Graph basics (3)

- ***Graph shaping***
  - ***Tensors*** is a mapping from a set of coordinates to a set of scalar values of the same data type
    - They are implemented in the form of multidimensional arrays
    - Correspond to the data in the data-flow graph
  - ***Ops*** are a composition of tensor transforms
    - The operations are nodes of the data-flow graph
    - The output of each operation has a fixed data type of elements and a dimension that corresponds to the dimension of the tensors in the graph
    - Inputs and outputs can be several
    - Each operation is a node of the data-flow graph, but not all nodes are operations
    - Some nodes of the graph describe the correspondence of sub-graphs in the process of optimizing the initial graph of computations

Graph basics (4)

- ***Graph shaping***
  - ***Functions*** group a set of operations
    - The function describes the sequence of operations on the input tensor
    - Operations can be performed on one memory location
    - The function definition includes the following steps:
      - Determination of parameters – tensors. When creating a tensor, the type of its elements and the size have be defined
      - The definition of nodes – operations performed on input tensors

Conclusion

- nGraph can be targeted for programming and deploying deep learning applications on the most modern compute devices
- Freedom to design user-facing APIs for various hardware deployments directly into the framework
- nGraph contains bridges for some of the more widely-known frameworks (the bridge acts as an intermediary between the ngraph core and the framework, providing a means to use various execution platforms)
- The tool is still under development, documentation is available on the official page [http://ngraph.nervanasys.com]

Literature

- Haykin S. Neural Networks: A Comprehensive Foundation. – Prentice Hall PTR Upper Saddle River, NJ, USA. – 1998.
- Osovsky S. Neural networks for information processing. – 2002.
- Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].

Authors

- **Kustikova Valentina Dmitrievna**
  Phd, lecturer, department of Computer software and supercomputer technologies, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  valentina.kustikova@itmm.unn.ru
- **Zhiltsov Maxim Sergeevich**
  master of the 1st year training, Institute of Information Technology, Mathematics and Mechanics, Nizhny Novgorod State University
  zhiltsov.max35@gmail.com
- **Zolotykh Nikolai Yurievich**
  D.Sc., Prof., department of Algebra, geometry and discrete mathematics, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  nikolai.zolotykh@itmm.unn.ru